Case Study: AWS S3 Full-Stack Image Processing Application

Project Brief

The goal of this project was to develop a scalable and efficient image processing application utilizing modern web technologies and cloud services. The application needed to:

- Allow users to upload images through a web interface.
- Dynamically display images stored in the cloud.
- Automate image resizing upon upload.
- Be hosted on a reliable cloud platform for scalability and availability.

This project showcases my expertise in building full-stack applications and deploying them to a cloud-based infrastructure.

Project Overview

This full-stack image processing application consists of a frontend for user interaction, a backend API for handling requests, and serverless automation for image resizing. The solution leverages AWS services such as S3, EC2, and Lambda to ensure scalability, performance, and cost-effectiveness.

Solution

Frontend

- **Technology:** HTML, CSS, JavaScript
- **Hosting:** AWS S3 Static Website Hosting

The frontend provides a user-friendly interface for:

1. **Uploading Images:** Users can select and upload images from their local device.
2. **Viewing Images:** Displays a list of images stored in the S3 bucket, allowing users to preview and select images dynamically.

Key Features:

- Interactive design for seamless user experience.

- Fetch API integration to communicate with the backend.
- Deployed on AWS S3 with public access for static website hosting.

Backend

- **Technology:** Node.js, Express.js
- **Hosting:** AWS EC2

The backend provides RESTful APIs to handle:

1. **Image Uploads:** Stores uploaded images in the `original-images/` folder within the S3 bucket.
2. **Image Retrieval:** Fetches a list of stored images and serves them to the frontend.

Key Features:

- AWS SDK integration for S3 operations.
- CORS configuration to support secure cross-origin requests.
- Deployed on an EC2 instance with security groups configured for public access.

Image Resizing Workflow

- **Technology:** AWS Lambda, Node.js (Sharp Library)
- **Trigger:** S3 Event Notifications

The image resizing process is automated using AWS Lambda:

1. When a new image is uploaded to the `original-images/` prefix, an S3 event triggers the Lambda function.
2. The function resizes the image to 300x300 pixels using the Sharp library.
3. The resized image is stored in the `resized-images/` prefix within the same bucket.

Key Features:

- Serverless architecture for cost efficiency and scalability.
- Optimized performance using native libraries for image processing.
- Seamless integration with S3 event notifications.

Challenges and Solutions

1. Cross-Origin Resource Sharing (CORS) Issues

- **Challenge:** Requests from the S3-hosted frontend to the EC2 backend were blocked by the browser due to CORS restrictions.
- **Solution:** Configured CORS middleware in the Express backend to allow secure communication between the two domains.

2. Lambda Runtime Compatibility

- **Challenge:** The Sharp library required a runtime environment compatible with AWS Lambda's Amazon Linux 2.
- **Solution:** Built a deployment package for Sharp using Docker to ensure compatibility with Lambda's environment.

3. Overlapping S3 Event Notifications

- **Challenge:** Conflicts in S3 event notification configurations caused ambiguous behavior.
- **Solution:** Consolidated rules and ensured distinct prefixes for triggers.

Results

- **Efficiency:** Images are resized automatically upon upload, streamlining the process.
- **Scalability:** Leveraged AWS services to ensure the application can handle high user demand.
- **User Experience:** Delivered a responsive and intuitive interface for uploading and viewing images.
- **Reliability:** Deployed a cloud-native solution with high availability and minimal maintenance.

Impact

This project highlights the following skills and achievements:

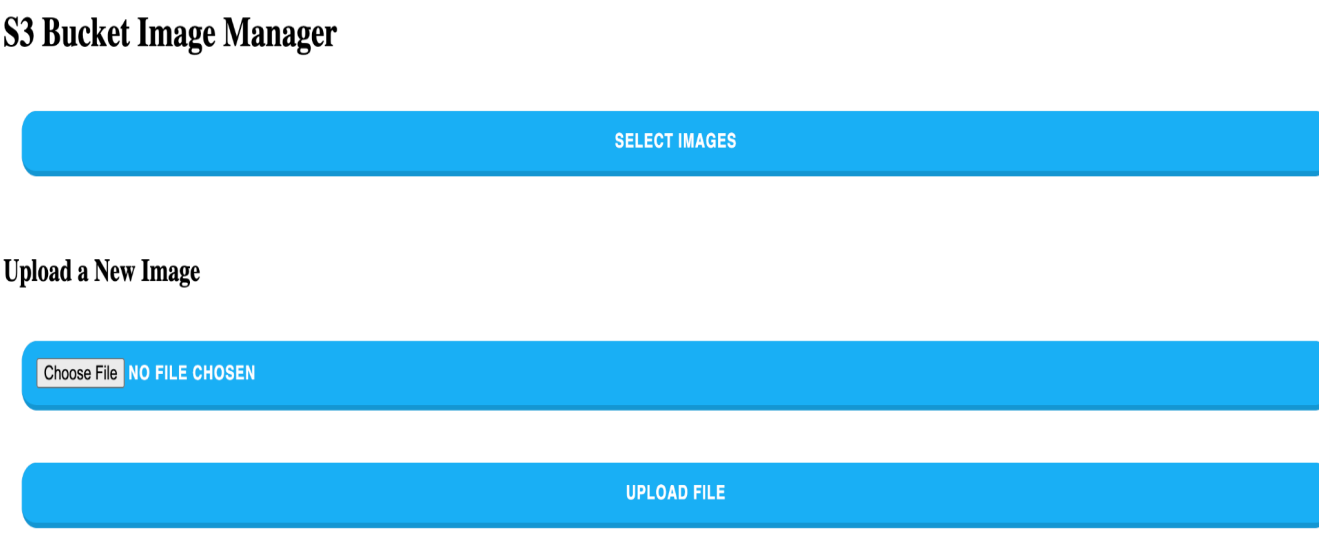
1. **Cloud Expertise:** Demonstrated proficiency in AWS services, including S3, EC2, and Lambda.
2. **Full-Stack Development:** Built a seamless integration of frontend and backend with modern tools and best practices.
3. **Problem-Solving:** Addressed real-world challenges such as CORS issues, serverless deployment, and event configuration conflicts.
4. **Automation:** Automated a critical workflow using serverless architecture, reducing manual intervention and enhancing scalability.

Conclusion

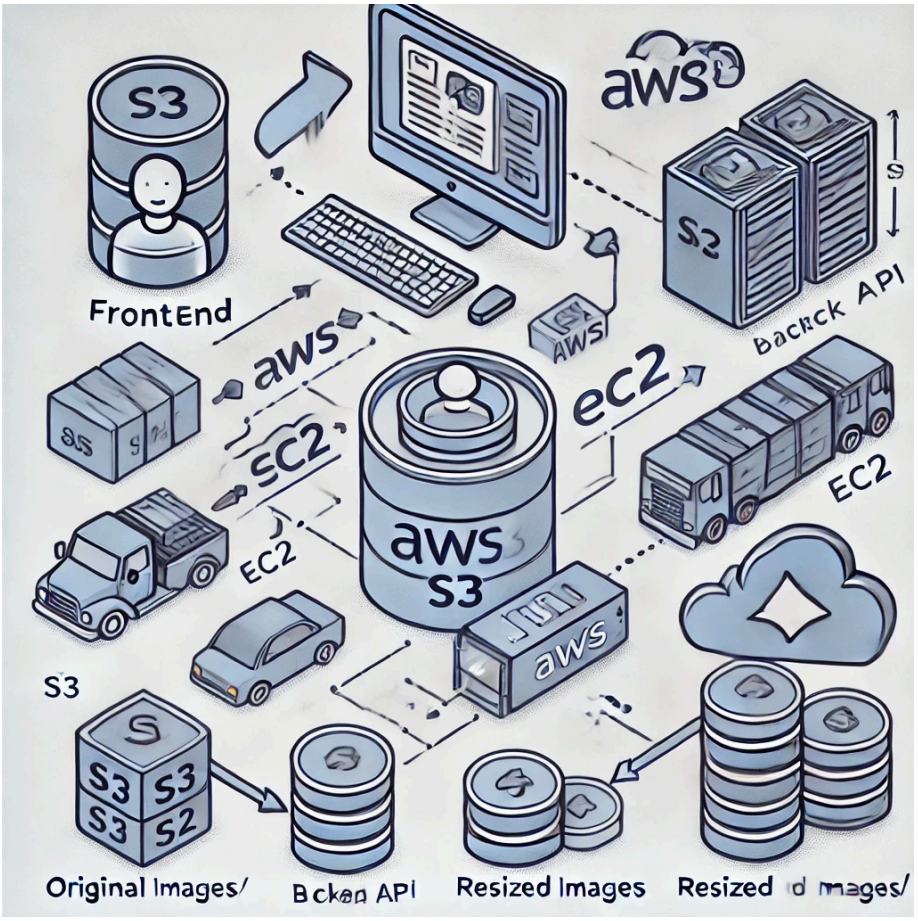
This project showcases my ability to design and deploy a complete cloud-based application, combining frontend, backend, and serverless technologies. It is a testament to my technical skills, problem-solving abilities, and commitment to delivering high-quality, scalable solutions.

Visuals

- Screenshots of the user interface (upload form, image gallery).



- Architecture diagram showcasing the integration of S3, EC2, and Lambda.



Next Steps

If you'd like to learn more about this project or discuss how my skills can benefit your team or project, feel free to reach out!